BAB III

METODOLOGI PENELITIAN

3.1 Analisa Masalah

Analisa keamanan algoritma RSA menggunakan metode fermats faktorisasi dan pollards rho bertujuan untuk mengetahui celah keamanan yang ada pada algoritma RSA yang dibangun dengan ketentuan $n = p < q < 2p$. Pengujian pada algoritma RSA perlu dilakukan sebagai parameter penelitian dan untuk mengetahui kelemahan yang terdapat pada algoritma ini. Sistem program yang dilakukan pada penelitian ini menerapkan metode fermat's faktorisasi dan pollard's rho untuk memfaktorkan nilai n dalam mendapatkan bilangan prima p dan q. Tingkat keamanan algoritma RSA sendiri dapat ditinjau dari panjang kunci publik n dan lama waktu memfaktorkan kunci publik n besar. Metode pengujian yang akan dilakukan dalam pemfaktoran kunci publik pada penelitian ini menggunakan metode fermat's faktorisasi dan pollard's rho untuk menemukan faktor-faktor primanya p dan q. Masalah yang terdapat dalam menganlisa keamanan algoritma RSA menggunakan metode femart's faktorisasi dan pollard's Rho adalah bagaimana usaha menguraikan suatu kunci publik kedalam faktor-faktor penyusunnya yaitu (p dan q) yang sebelumnya sudah dibangkitkan dengan ketentuan $n = p < q < 2p$ untuk mempersulit dalam menemukan nilai p dan q.

3.2 Perancangan program

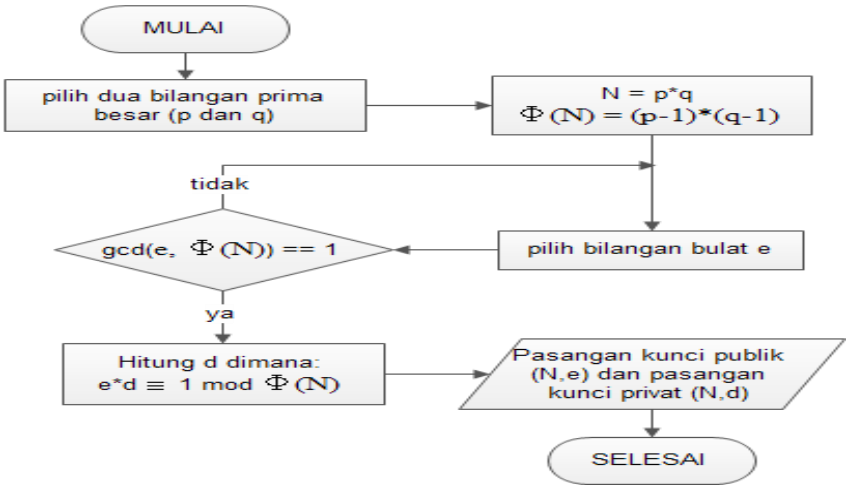
Program yang dibuat pada penelitian ini menggunakan program berbahasa java yang nantinya akan mengimplementasikan algoritma RSA, metode femart's faktorisasi dan pollard's rho sebagai metode faktorisasi.

3.2.1 Rancangan Algoritma RSA Standar

Membangun algoritma RSA untuk menghasilkan kunci publik dan kunci privat yang dibuat dengan membangkitkan kunci RSA untuk mendapatkan kunci publik dan kunci privat yang memenuhi $n = p < q < 2p$. Lebih jelasnya kita lihat proses pembangkitan kunci nya.

3.2.1.1 Pembangkitan Kunci Algoritma RSA Standar

1. Flow Chart Pembangkitan Kunci RSA



Gambar 3.1 : Flow chart proses pembangkitan kunci

2. Contoh Perhitungan Manual Proses Pembangkitan Kunci Publik

Menentukan bilangan prima yang akan di pilih $n = p < q < 2p$. Proses perhitungan manual dalam membangkitkan kunci publik diibaratkan oleh user A, langkah-langkah yang akan dilakukan A antara lain sebagai berikut:

1) Menentukan bilangan prima yang akan dipilih, $n = p < q < 2p$ dimana nilai $p = 17\ dan\ q = 23$
2) Menggunakan rumus 2.1 untuk menghitung nilai $n = 17 * 23 = 391$.
3) Menggunakan rumus 2.2 untuk menghitung phi-n dimana $\Phi(n) = (17 - 1) * (23 - 1) = 16 * 22 = 352$.

4) Menggunakan rumus 2.3 untuk mencari kunci publik $e = 5$ relatif prima dengan rumus, selanjutnya mengumumkan nilai *e* dan n yaitu $\gcd(5, 352) = 1$.
5) Menggunakan rumus 2.4 untuk menghitung kunci privat $d = 141$, $5 * 141 \equiv 1\ mod\ 352$.
6) Perhitungan yang telah dilakukan di atas akan menghasilkan pasangan kunci publik *(391, 5)* dan pasangan kunci privat *(391, 141).* Kunci privat digunakan untuk mendekripsi pesan .

3. Pseudo – Code Pembangkitan Kunci Algoritma RSA

```
Function pembangkitankunci()
 #menentukan p dan q dari panjang bit
 #hitung nilai n
    n = p*q (n = p < q < 2p)
    phi_n = (p-1)*(q-1)
 #membaca bilangan bulat e
    e = gcd(e, phi_n)) = 1, 1 < e < phi_n
    if 1 == gcd (e, phi_n)
      d = e mod phi_n
      return (e,d), (e, n);
 end
```

Gambar 3.2 : PseudoCode Pembangkitan Kunci Algoritma RSA

4. Penjelasan PseudoCode Pembangkitan Kunci Algoritma RSA

Langkah pertama yang dilakukan adalah membaca dua bilangan prima BigInteger (untuk menyimpan bilangan bulat besar) p dan q. Setelah itu, kita dapat menghitung nilai n dengan rumus 2.1 dilanjut menghitung nilai phi_n dengan rumus 2.2. Kunci publik didapatkan dari mengghitung nilai e dari rumus 2.3 dan kunci privat dapat di cari berdasarkan nilai d degngan rumus 2.4. Output yang dihasilkan dari pembangkitan kunci di atas adalah nilai (n, e dan n,d).

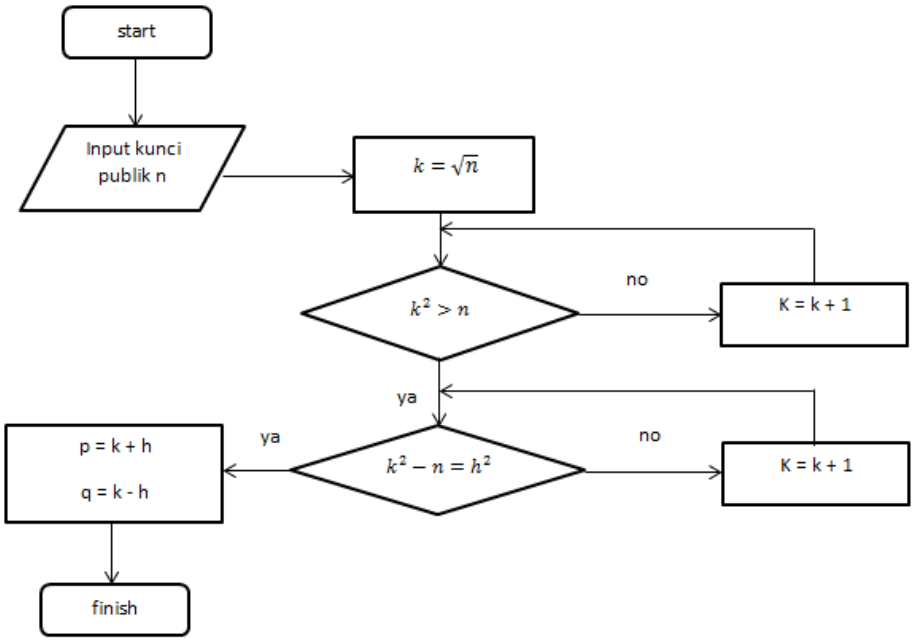
3.3 Rancangan Algoritma Fermats Faktorisasi

Membangun algoritma fermats faktorisasi untuk melakukan proses faktorisasi pada kunci publik yang sudah dibangkitkan, program akan dijalankan di sebuah

program java, dengan itu kunci publik dapat di analisa keamanannya lewat faktorisasi kunci publik.

3.3.1 Metode Femart Faktorisasi

1. Flow Chart Metode Fermat Faktorisasi *(Segar & Vijayaragavan, 2013)*



Gambar 3.3 : Flow chart proses faktorisasi algoritma fermat's faktorisasi

2. Contoh Perhitungan Manual Proses Faktorisasi Menggunakan Algoritma Fermat Faktorisasi.

Berikut ini adalah contoh manual dari proses faktorisasi algortima *fermat's faktorisasi* untuk mendapatkan bilangan prima yang akan di terapkan di algoritma RSA. Sebagai contoh kita coba dengan nilai $n = 391$:

Langkah 1 :

Menggunakan rumus dari 2.5 dimana $n = \sqrt{391},\ n = 19$

Langkah 2 :

Menggunakan rumus dari 2.6 menghitung nilai k^2

$k^2 = 19^2 = 361 < 391$ masih kurang dari niali n maka +1

$k = 20,\ k^2 = 400 > 391$ sudah memenuhi maka lanjut

Langkah 3 :

Menggunakan rumus dari 2.7 nilai dari h^2 harus akar kuadrat sempurna jadi $h^2 = 20^2 - 391 = 9$, $\sqrt{9} = 3 \rightarrow k^2 = 3$ kuadrat sempurna maka lanjut, jika belum maka k+1.

Langkah 4 :

Menggunakan rumus dari 2.8 menghitung nilai faktor p dan q jadi $p = 20 + 3 = 23, q = 20 - 3 = 17$.

Hasil yang didapatkan menggunakan rumus 2.8 yaitu $20 + 3 = 23$ dan $20 - 3 = 17$, kemudian setelah itu dapatkan dengan rumus 2.1 yaitu $23 * 17 = 391$. Apabila suatu bilangan komposit n merupakan produk dari dua bilangan prima p dan q yang merupakan bilangan prima ganjil yang berdekatan. Bisa dikatakan n bukan merupakan produk dari dua bilangan ganjil yang berdekatan, maka *Fermat Faktorisasi* akan mencoba banyak nilai k sebelum menemukan nilai k untuk mendapatkan factor-faktor dari n.

3. PseudoCode Algoitma Femart's Faktorisasi

```
Function fermatfaktorisasi (n)
   #input nilai kunci publik (n)
      For k from ceil (sqrt (n)) to n
      h square = k * k-n
      if p > 1 and p < n do
         h = sqrt (hSquared)
         p = k + h
         q = k – h
      return p, q
```

Gambar 3.4: PseudoCode Algoritma Fermat's Faktorisasi

4. Penjelasan PseuduCode Algoritma Fermat's Faktorisasi

Menginput nilai n, nilai n di faktorisasi untuk mendapatkan nilai p dan q. Nilai n di cek terlebih dahulu oleh k apakah n merupakan kuadrat sempurna, setelah menggetahui k merupakan kuadrat sempurna maka dilihat lagi apakah k lebih besar daripada n. Perhitungan selanjutnya dapat dilakukan apabila nilai k lebih besar dari nilai n, jika (ya) maka lanjut

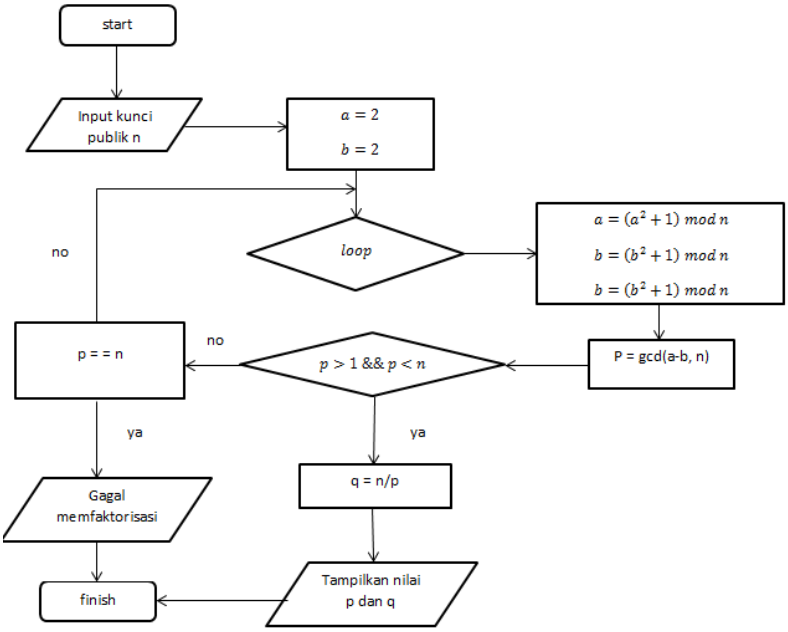
dengan menghitung rumus dari 2.7 sampai hasil dari k adalah kuadrat sempurna jika tidak maka ditambahkan nilai 1 pada k. Langkah selanjutnya setelah mendapat kuadrat sempurna dari k, kita cari nilai p dengan menghitung dari rumus 2.8, sedangkan nilai q di dapat dengan menghitung dari rumus 2., sehingga kita dapatkan nilai p dan q tetapi apabila $p > 1 \; dan \; p < n$ maka kita dapatkan nilai p dan q.

3.4 Rancangan Metode Algoritma Pollard Rho

Membangun algoritma pollard rho untuk melakukan proses faktorisasi pada kunci publik yang sudah dibangkitkan, program akan dijalankan di sebuah program java, dengan itu kunci publik dapat di analisa keamanannya lewat faktorisasi kunci publik.

3.4.1 Metode Pollard-rho

1. Flow Chart Metode Pollard Rho *(Jaybhay and Jadhav, 2015).*



Gambar 3.5 : Flow chart proses faktorisasi algoritma Pollard-Rho

2. Contoh Perhitungan Manual Proses Faktorisasi Menggunakan Algoritma Pollard Rho.

Berikut ini adalah contoh manual dari proses faktorisasi algortima *pollard-rho* untuk mendapatkan bilangan prima yang akan di terapkan di algoritma RSA. Masukan nilai *n = 391*, selanjutnya akan di proses berdasarkan algoritma *pollard'd rho* untuk mencari nilai faktor.

Iterasi 1 :

menggunakan rumus dari 2.9 menetukan nilai a dan b

$a = 2, b = 2$

menggunakan rumus dari 2.10 untuk menghitung nilai a

$a = (2^2 + 1)\ mod\ 391$

$= (5 + 1)\ mod\ 3391 = 5$

menggunakan rumus dari 2.10 menghitung nilai b

$b = (2^2 + 1)\ mod\ 391$

$= (5 + 1)\ mod\ 391 = 5$

Hitung ulang b

$b = (5^2 + 1) mod\ 391$

$= 26\ mod\ 391 = 26$

menggunakan rumus dari 2.11 menghitung faktor p

$p = \gcd(5 - 26, 391)$

$p = \gcd(-21, 391) = 1$

Iterasi 2 :

menggunakan rumus dari 2.9 menentukan nilai a dan b

$a = 5, b = 26$

menggunakan rumus dari 2.10 untuk menghitung nilai a

$a = (5^2 + 1)\ mod\ 391$

$a = 26\ mod\ 391 = 26$

.menggunakan rumus dari 2.10 menghitung nilai b

$b = (26^2 + 1) mod\ 391$

$b = 677\ mod\ 391 = 286$

Hitung ulang b

$$b = (286^2 + 1) mod\ 391$$

$$b = 76648\ mod\ 391 = 78$$

menggunakan rumus dari 2.11 mencari faktor p

$$p = \gcd(26 - 78, 391)$$

$$p = \gcd(-52, 391) = 1$$

iterasi 3 :

menggunakan rumus dari 2.9 menentukan nilai a dan b

$$a = 26, b = 78$$

menggunakan rumus dari 2.10 untuk menghitung nilai a

$$a = (26^2 + 1) mod\ 391$$

$$a = 677\ mod\ 391 = 286$$

menggunakan rumus dari 2.10 menghitung nilai b

$$b = (78^2 + 1) mod\ 391$$

$$b = 6085\ mod\ 391 = 220$$

Hitung ulang b

$$b = (220^2 + 1) mod\ 391$$

$$b = 48401\ mod\ 391 = 308$$

menggunakan rumus dari 2.11 mencari faktor p

$$p = \gcd(286 - 308, 391)$$

$$p = \gcd(-22, 391) = 1$$

iterasi 4 :

menggunakan rumus dari 2.9 menentukan nilai a dan b

$$a = 286, b = 308$$

menggunakan rumus dari 2.10 untuk menghitung nilai a

$$a = (286^2 + 1) mod\ 391$$

$$a = 81797\ mod\ 391 = 78$$

menggunakan rumus dari 2.10 menghitung nilai b

$$b = (308^2 + 1) mod\ 391$$

$$b = 94861\ mod\ 391 = 243$$

Hitung ulang b

$$b = (243^2 + 1) mod\ 391$$

$$b = 59050\ mod\ 391 = 17$$

menggunakan rumus dari 2.11 mencari faktor p

$$p = \gcd(78 - 17)$$

$$p = \gcd(61, 391) = 23$$

Tabel 3.1 : Output dari proses iterasi metode pollards rho

Total iterasi	A	B	p = [gcd (a-b)]
1	5	26	1
2	26	78	1
3	286	308	1
4	78	9	23

Nilai yang di dapatkan yaitu 23 dimana hasil tersebut didapatkan dari faktor *n = 391*, selanjutnya menggunakan rumus 2.12 kemudian nilai p = 23 dibagi dengan nilai n untuk mendapatkan faktor q, *q = 391 : 23 q = 17*, sehingga didapatkan faktor dari *n = 391* yaitu nilai *p = 23* dan *q = 17*.

3. PesudoCode Algoritma Pollard Rho

```
Fungtion pollard_rho (n)
    #input kunci public (n)
        Count velueOf    a(2), b(2), and b(2);
          while (true) a(a^2 + 1), b (b^2 +1(b^2 + 1));
             Count p = (a -b), gcd (n);
                Println ( p) ;
                   Loop (a,b);
                   False if (p = n);
            If p > 1 and p < n than
             Count q = (n/p);
                Println (q);
```

Gambar 3.6: PseudoCode Algoritma Pollard Rho

4. Penjelasan Pesudo Code Algoritma Pollard Rho

Memfaktorisasi nilai n menggunakan metode pollard rho yaitu pertama menginput kunci publik n berdasarkan rumus dari 2.1 untuk memfaktorkanya. Langkah selanjutnya menghitung nilai a dan b dimana nilai a dan b bernilai masing-masing 2 berdasarkan rumus dari 2.10, setelah selesai menghitung nilai a dan b kemudian menghitung nilai p menggunakan rumus 2.1 untuk mendapatkan nilai p yang sebelumnya nilai p di cek terlebih dahulu apakah nilai p sudah memenuhi syarat $p > 1\ dan\ p < n$ jika tidak memenuhi syarat tersebut atau nilai $p = n$ maka dilakukan perhitungan ulang di rumus 2.10. Langkah selanjutnya apabila nilai p sudah memenui syarat $p > 1\ dan\ p < n$ maka nilai q dapat di hitung dengan rumus 2.12.